# Projektthema: Kalender

## Erfassung der Kundenbedürfnisse

### Initialbesprechung

Der Kunde möchte ein maßgeschneidertes Kalender System. Es soll interne Veranstaltungen, Meetings und Projekte speichern. Einfache Bedingung ist dem Kunden wichtig.   
Das Sortieren nach Typen, Benutzern und Zeiten soll ermöglicht werden.  
Der Kalender soll auch über mobile Plattformen erreichbar sein.  
Die Daten sollen sicher sein.

App soll innerhalb von drei Tagen entwickelt werden!

### Marktforschung

Machen wir jetzt mal nicht

### Technische Prüfung

HTML und Bootstrap für die Frontend-Entwicklung:

Vorteile: Mit HTML und Bootstrap können wir schnell eine responsive Benutzeroberfläche entwickeln, die auf verschiedenen Geräten und Bildschirmgrößen gut funktioniert. Bootstrap bietet eine Reihe von vordefinierten Komponenten und Layouts, die die Entwicklung beschleunigen. Wichtig, da wir nur sehr wenig Zeit haben.

Nachteile: Bei komplexeren Anpassungen könnten die Limitierungen von Bootstrap in Bezug auf Design und Funktionalität zum Tragen kommen. Wir müssen darauf achten, dass die Anwendung nicht zu abhängig von Bootstrap-spezifischen Klassen und Strukturen wird, um Flexibilität in der Zukunft zu gewährleisten.

CSS für die Gestaltung:

Vorteile: CSS ermöglicht es uns, das Design und das Look-and-Feel der Kalenderanwendung genau nach den Wünschen des Kunden anzupassen. Mit modernen Techniken wie Flexbox und Grid können wir ein flexibles und ansprechendes Layout erstellen.

Nachteile: Die Komplexität des Designs kann die Wartung erschweren, besonders wenn wir viele individuelle Anpassungen vornehmen. Wir müssen eine gut organisierte Struktur der Stylesheets sicherstellen.

MariaDB als Datenbanksystem:

Vorteile: MariaDB bietet eine leistungsstarke, skalierbare und kompatible Alternative zu MySQL. Es unterstützt komplexe Abfragen, Transaktionssicherheit und ist gut für die Speicherung der Datenstrukturen geeignet, die unser Kalender-System benötigt.

Nachteile: Die Migration von bestehenden Datenbanken oder die Integration mit Systemen, die nicht optimal mit MariaDB zusammenarbeiten, könnte Herausforderungen darstellen. Wir müssen sicherstellen, dass alle nötigen Sicherheits- und Performance-Optimierungen durchgeführt werden.

PHP für das Backend:

Vorteile: PHP ist weit verbreitet für die Entwicklung von Webanwendungen. Es bietet eine große Menge an Bibliotheken und Frameworks, die die Entwicklung beschleunigen können. PHP ist besonders gut geeignet für die Erstellung dynamischer Inhalte und die Integration mit MariaDB.

Nachteile: PHP-Anwendungen können bei schlechter Architektur anfällig für Sicherheitsprobleme sein. Wir müssen Best Practices für die PHP-Entwicklung befolgen und das System regelmäßig auf Sicherheitslücken überprüfen.

## Dokumentation der Anforderungen

Anforderungsdokument für das Kalender-Projekt

Projektbezeichnung: Entwicklungsprojekt des Unternehmenskalender-Systems

Datum: 18. März 2024

Version: 1.0

1. Projektübersicht

Das Kalender-Projekt zielt darauf ab, ein zentrales, webbasiertes Kalendersystem, das die Planung und Verwaltung von internen Veranstaltungen, Meetings und Projektfälligkeiten vereinfacht. Das System soll eine intuitive Benutzeroberfläche bieten, von überall zugänglich sein und eine Integration mit bestehenden Kommunikations- und Kollaborationstools ermöglichen.

2. Stakeholder

Auftraggeber: [Name des Kundenunternehmens]

Projektmanager: [Name]

Entwicklungsteam: [Namen der Teammitglieder]

Endbenutzer: Mitarbeiter von [Kundenunternehmen]

3. Funktionale Anforderungen

Veranstaltungsmanagement:

Erstellung, Bearbeitung und Löschung von Veranstaltungen.

Unterscheidung verschiedener Veranstaltungstypen (Meetings, Deadlines, Firmenevents).

4. Zugänglichkeit und Responsive Design:

Vollständig responsive Webanwendung, zugänglich auf Desktop- und mobilen Endgeräten.

5. Nicht-funktionale Anforderungen

Performance:

Schnelles verarbeiten (<2 Sekunden)

Sicherheit:

Verschlüsselte Datenübertragung

6. Technologieauswahl

Frontend: HTML5, CSS3 (mit Bootstrap), JavaScript

Backend: PHP 7.x

Datenbank: MariaDB

Sicherheit: ?

**Tag 1: Planung und Setup**

Vormittag - Anforderungsanalyse und Planung:

Detaillierte Besprechung der Kundenanforderungen, um Missverständnisse zu vermeiden.

Definition der Kernfunktionalitäten, die in der MVP-Version enthalten sein sollen: Ereigniserstellung, Sortierung und mobile Zugänglichkeit.

Auswahl der Technologien: Angesichts der kurzen Entwicklungszeit und der Anforderungen empfehlen wir die Verwendung von Frameworks und Tools, die eine schnelle Entwicklung ermöglichen.

Nachmittag - Projektsetup:

Einrichtung der Entwicklungsumgebung und des Repositories (z.B. GitHub).

Initialisierung der Projekte mit den gewählten Frameworks.

Setup der Datenbankstruktur in MariaDB. Fokussierung auf eine einfache Struktur, die Veranstaltungen, Benutzer und Typen speichern kann.

**Tag 2: Kernfunktionalität und Basis-UI**

Vormittag - Backend-Entwicklung:

Implementierung der API-Endpunkte für das Erstellen, Abrufen, Aktualisieren und Löschen von Veranstaltungen.

Einrichtung der Authentifizierung, um die Sicherheit der Daten zu gewährleisten.

Erstellung von Logik zur Sortierung von Daten nach Typen, Benutzern und Zeiten.

Nachmittag - Frontend-Entwicklung:

Erstellung einer einfachen, aber responsiven Benutzeroberfläche, die auf Mobilgeräten gut funktioniert. Verwendung von Bootstrap oder ähnlichen CSS-Frameworks, um die Entwicklung zu beschleunigen.

Integration der Frontend-Anwendung mit den Backend-APIs.

Einfache Tests auf verschiedenen Geräten, um die mobile Zugänglichkeit zu gewährleisten.

**Tag 3: Feinabstimmung, Sicherheit und Übergabe**

Vormittag - Feinabstimmung und Tests:

Durchführung von Tests, um sicherzustellen, dass die Kernfunktionen wie erwartet funktionieren.

Anpassungen basierend auf Testergebnissen, um Bugs zu beheben und die Benutzererfahrung zu verbessern.

Nachmittag - Sicherheitsüberprüfungen und Dokumentation:

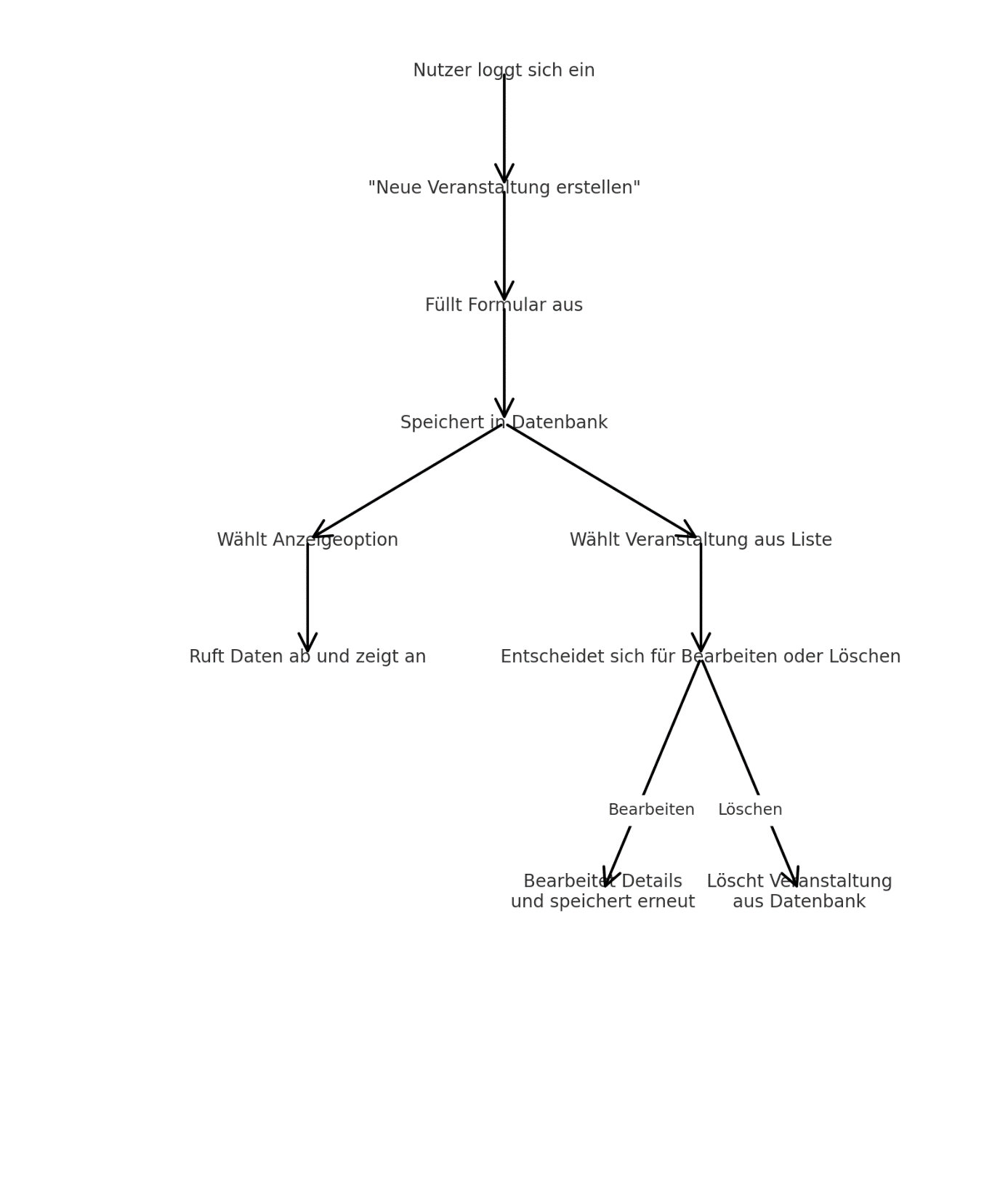
Überprüfung der Sicherheitsaspekte, einschließlich der Datenübertragung und -speicherung. Implementierung von HTTPS, Nutzung sicherer Authentifizierungspraktiken.

Erstellung einer einfachen Dokumentation für die Benutzung der App und eine Übersicht der Architektur.

## 2.Modellierung

### 2.1 Prozessmodell

1. **Veranstaltungserstellung:**
   * Der Nutzer loggt sich ein und wählt die Option „Neue Veranstaltung erstellen“.
   * Der Nutzer füllt ein Formular mit Details wie Titel, Beschreibung, Typ (Meeting, internes Event, Projektdeadline), Start-/Endzeit und Beteiligte aus.
   * Das System speichert die Veranstaltung in der Datenbank.
2. **Veranstaltungsanzeige:**
   * Der Nutzer wählt, ob er alle Veranstaltungen anzeigen möchte oder eine gefilterte Ansicht nach Typ, Zeitraum oder beteiligten Benutzern bevorzugt.
   * Das System ruft die entsprechenden Daten aus der Datenbank ab und zeigt sie an.
3. **Veranstaltungsaktualisierung/-löschung:**
   * Der Nutzer wählt eine bestehende Veranstaltung aus der Liste aus und entscheidet sich für Bearbeiten oder Löschen.
   * Bei der Bearbeitung kann der Nutzer die Veranstaltungsdetails anpassen und speichern.
   * Beim Löschen wird die Veranstaltung aus der Datenbank entfernt.



### 2.2 Benutzeroberfläche

* **Login-Seite:** Ein einfaches Formular für Benutzername und Passwort.
* **Hauptansicht des Kalenders:** Eine Monats-/Wochen-/Tagesansicht, die die geplanten Veranstaltungen zeigt. Ein Button für „Neue Veranstaltung erstellen“ ist prominent platziert.
* **Veranstaltungsformular:** Ein Formular zum Eingeben der Details einer neuen oder zu bearbeitenden Veranstaltung. Felder umfassen Titel, Beschreibung, Typ, Datum/Uhrzeit und Beteiligte.
* **Filteroptionen:** Dropdown-Menüs oder Schaltflächen, um die Ansicht nach Typ, Zeitraum oder beteiligten Benutzern zu filtern.

### 2.3 Datenmodell

* **Benutzer:**
  + ID (Primärschlüssel)
  + Benutzername
  + Passwort (verschlüsselt gespeichert)
  + E-Mail
* **Veranstaltungen:**
  + ID (Primärschlüssel)
  + Titel
  + Beschreibung
  + Typ (Fremdschlüssel zu Veranstaltungstypen)
  + Startzeit
  + Endzeit
  + Ersteller (Fremdschlüssel zu Benutzer)
* **Veranstaltungstypen:**
  + ID (Primärschlüssel)
  + Typname (z.B. Meeting, Internes Event, Projektdeadline)

**Ein Bild, das Screenshot, Text, Diagramm, Reihe enthält.

Automatisch generierte Beschreibung**

-- Benutzer (User)

CREATE TABLE IF NOT EXISTS Benutzer (

ID INT AUTO\_INCREMENT PRIMARY KEY,

Benutzername VARCHAR(255) NOT NULL,

Passwort VARCHAR(255) NOT NULL,

Email VARCHAR(255) NOT NULL

);

-- Veranstaltungstyp (Event Type)

CREATE TABLE IF NOT EXISTS Veranstaltungstyp (

ID INT AUTO\_INCREMENT PRIMARY KEY,

Name VARCHAR(255) NOT NULL

);

-- Veranstaltung (Event)

CREATE TABLE IF NOT EXISTS Veranstaltung (

ID INT AUTO\_INCREMENT PRIMARY KEY,

Titel VARCHAR(255) NOT NULL,

Beschreibung TEXT,

Startzeit DATETIME NOT NULL,

Endzeit DATETIME NOT NULL,

Typ\_ID INT,

Benutzer\_ID INT,

FOREIGN KEY (Typ\_ID) REFERENCES Veranstaltungstyp(ID),

FOREIGN KEY (Benutzer\_ID) REFERENCES Benutzer(ID)

);

## 3.Frontend-Entwicklung (HTML & CSS)

## HTML:

* Erstellen von HTML-Dokumenten für jede Seite der Anwendung (z.B. Login-Seite, Kalenderansicht, Veranstaltungsformular).
* Anwendung von semantischem HTML, um die Barrierefreiheit und SEO zu verbessern.
* Integration von Frontend-Frameworks wie Bootstrap für responsive Design und vorgefertigte Komponenten.

## CSS:

* Erstellung von CSS-Stylesheets zur Definition des visuellen Erscheinungsbilds.
* Anwendung von Medienabfragen, um eine reaktionsschnelle Oberfläche zu gewährleisten, die auf verschiedenen Geräten gut funktioniert.

## PHP:

* Einrichtung der Serverumgebung und der PHP-Konfiguration.
* Entwicklung von PHP-Skripten, um Anfragen zu verarbeiten, mit der Datenbank zu interagieren und dynamische Inhalte zu generieren.
* Anwendung von objektorientiertem PHP für eine saubere und wartbare Codebasis.

## Datenbank-Integration:

* Verwendung von PHP's PDO für sichere Datenbankverbindungen und Anfragen.
* Entwicklung von CRUD-Operationen (Create, Read, Update, Delete) für die Verwaltung von Veranstaltungen.
* Implementierung von Authentifizierungs- und Autorisierungslogiken für Benutzer.

## 3.3 Verbindung von Frontend und Backend

* Formular-Submits werden verwendet, um Daten vom Frontend an das Backend zu senden und Antworten zurückzubekommen.
* Nutzung von PHP-Sessions für die Benutzerverwaltung und -session.

## 3.4 Testing und Debugging

* Durchführung von manuellen Tests der Benutzeroberfläche, um die Benutzererfahrung zu überprüfen.
* Schreiben von Unit-Tests für die PHP-Funktionen, um Logik und Datenbankinteraktionen zu testen.

## 3.5 Deployment

* Vorbereitung der Anwendung für das Deployment, einschließlich der Konfiguration der Produktionsumgebung.
* Hochladen der Dateien auf einen Webserver oder eine Hosting-Plattform.
* Durchführung von End-to-End-Tests, um die Funktionalität in der Produktionsumgebung zu bestätigen.

